3

Interrupt Processing

3.1 Introduction

Different types of processors respond to the occurence of an interrupt in its own unique fashion. In addition, each processor type provides a control mechanism to allow the proper handling of an interrupt. The processor dependent response to the interrupt which modifies the execution state and results in the modification of the execution stream. This modification usually requires that an interrupt handler utilize the provided control mechanisms to return to the normal processing stream. Although **RTEMS** hides many of the processor dependent details of interrupt processing, it is important to understand how the **RTEMS** interrupt manager is mapped onto the processor's unique architecture. Discussed in this chapter are the the processor's response and control mechanisms as they pertain to **RTEMS**.

3.2 Vectoring of Interrupt Handler

Upon receipt of an interrupt the i960CA automatically performs the following actions:

- *saves the local register set,*
- *sets the Frame Pointer (FP) to point to the interrupt stack,*
- *increments the FP by sixteen (16) to make room for the Interrupt Record,*

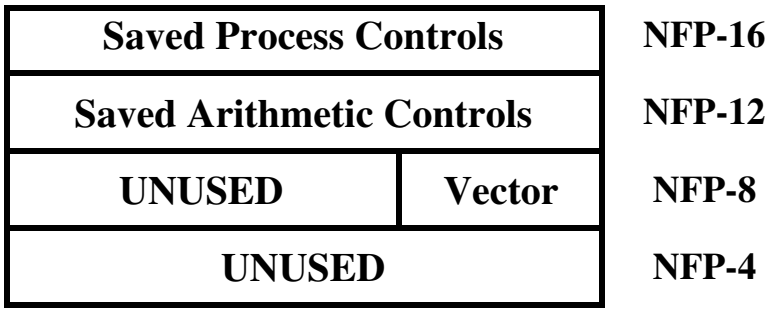
- *saves the current values of the arithmetic-controls (AC) register, the process-controls (PC) register, and the interrupt vector number are saved in the Interrupt Record,*
- *the CPU sets the Instruction Pointer (IP) to the address of the first instruction in the interrupt handler,*
- *the return-status field of the Previous Frame Pointer (PFP or R0) register is set to interrupt return,*
- *sets the PC state bit to interrupted,*
- *sets the current interrupt disable level in the PC to the level of the current interrupt, and*
- *disables tracing.*

A nested interrupt is processed similarly by the i960CA with the exception that the Frame Pointer (FP) already points to the interrupt stack. This means that the FP is **NOT** overwritten before space for the Interrupt Record is allocated.

The state flag bit of the saved PC register in the Interrupt Record is examined by **RTEMS** to determine when an outer most interrupt is being exited. Therefore, the user application code **MUST NOT** modify this bit.

3.3 Interrupt Record

The structure of the Interrupt Record for the i960CA which is placed on the interrupt stack by the processor in response to an interrupt is as follows:



3.4 Interrupt Levels

Thirty-two levels (0-31) of interrupt priorities are supported by the i960CA microprocessor with level thirty-one (31) being the highest priority. Level zero (0) indicates that interrupts are fully enabled. Interrupt requests for interrupts with priorities less than or equal to the current interrupt mask level are ignored.

Although **RTEMS** supports 256 interrupt levels, the i960CA only supports thirty-two. **RTEMS** interrupt levels 0 through 31 directly correspond to i960CA

interrupt levels. All other **RTEMS** interrupt levels are undefined and their behavior is unpredictable.

3.5 Disabling of Interrupts by RTEMS

During the execution of directive calls, critical sections of code may be executed. When these sections are encountered, **RTEMS** disables interrupts to level thirty-one (31) before the execution of this section and restores them to the previous level upon completion of the section. **RTEMS** has been optimized to insure that interrupts are disabled for less than 2.5 microseconds on a 33 Mhz i960CA with zero wait states. These numbers will vary based the number of wait states and processor speed present on the target board.

Non-maskable interrupts (NMI) cannot be disabled, and ISRs which execute at this level **MUST NEVER** issue **RTEMS** system calls. If a directive is invoked, unpredictable results may occur due to the inability of **RTEMS** to protect its critical sections. However, ISRs that make no system calls may safely execute as non-maskable interrupts.

3.6 Register Cache Flushing

The i960CA version of the **RTEMS** interrupt manager is optimized to insure that the **flushreg** instruction is only executed when a context switch is necessary. The **flushreg** instruction flushes the i960CA register set cache and takes (14 + 23 * number of sets flushed) cycles to execute. As the i960CA supports caching of from five to sixteen register sets, this instruction takes from 129 to 382 cycles (3.90 to 11.57 microseconds at 33 Mhz) to execute given no wait state memory. **RTEMS** flushes the register set cache only at the conclusion of the outermost ISR when a context switch is necessary. The register set cache will not be flushed as part of processing a nested interrupt or when a context switch is not necessary. This optimization is essential to providing high-performance interrupt management on the i960CA.

3.7 Interrupt Stack

On the i960CA, **RTEMS** allocates the interrupt stack from the **Workspace Area**. The amount of memory allocated for the interrupt stack is determined by the **interrupt_stack_size** field in the **CPU Configuration Table**. During the initialization process, **RTEMS** will install its interrupt stack.